Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Kristina Peteln

Razpoznavanje prevoznih sredstev na osnovi uporabe senzorjev pametnih mobilnih naprav

diplomsko delo
na univerzitetnem študiju

Mentor: doc. dr. Rok Rupnik

Ljubljana 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Zasnujte in razvijte mobilno aplikacijo za sprotno razločevanje med hojo, kolesarjenjem, vožnjo z avtomobilom, vožnjo z mestnim avtobusom in mirovanjem. To omogoča uporabniku prilagojeno analizo trajanja poti z različnimi prevoznimi sredstvi. Pri tem temeljite na uporabi senzorjev, predvsem pospeškometer in GPS. Za izdelavo modelov učenja iz podatkov uporabite logistično regresijo, Newtonovo metodo in gradientni spust. Mobilna aplikacija naj bo razvita za platformo Android.

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Kristina Peteln, z vpisno številko **63080151**, sem avtorica diplomskega dela z naslovom:

Razpoznavanje prevoznih sredstev na osnovi uporabe senzorjev pametnih mobilnih naprav

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom doc. dr. Roka Rupnika,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 5. julija 2016 Podpis avtorice:

Zahvaljujem se mentorju doc. dr. Roku Rupniku za dosegljivost in nasvete pri izdelavi diplomske naloge. Prav tako se zahvaljujem Mihi za podporo in družini, ki mi je med študijem stala ob strani.

Kazalo

Povzetek

Namen diplomskega dela je razvoj mobilne aplikacije za sprotno razločevanje med hojo, kolesarjenjem, vožnjo z avtomobilom, vožnjo z mestnim avtobusom in mirovanjem, kar omogoča uporabniku prilagojeno analizo trajanja poti z različnimi prevoznimi sredstvi. Za dosego cilja so bili zbrani podatki iz senzorjev, ki jih vsebujejo sodobne mobilne naprave (pospeškomer in GPS), iz katerih so bili s pomočjo logistične regresije naučeni modeli za klasifikacijo. Za računanje parametrov klasifikacije sta bila z različnimi argumenti uporabljena Newtonova metoda in gradientni spust. Modeli so bili uporabljeni v Android aplikaciji, ki sproti razločuje med prevoznimi sredstvi in poti shranjuje za kasnejšo analizo. Na voljo je tudi možnost naknadne korekcije napačno klasificiranih osamelcev in razločevanja med avtomobilom in avtobusom glede na postanke na poti.

Ključne besede: logistična regresija, Newtonova metoda, gradientni spust, Android aplikacija

Abstract

The purpose of this thesis is the development of a mobile application that can distinguish between walking, biking, driving by car, driving by bus and standing still, which enables the user to perform a personal analysis of their travel duration according to different means of transport. The objective was achieved by gathering accelerometer and GPS data from sensors included in modern mobile devices and using it to train classification models with logistic regression. The classification parameters were calculated using Newton's method and gradient descent with various arguments. The models were utilized in an Android application, which distinguishes between transportation modes in real time and saves the routes for later analysis. The application also provides options for subsequent correction of wrongly classified outliers and further distinction between bus and car based on stopovers.

Key words: logistic regression, Newton's method, gradient descent, Android application

Poglavje 1

Uvod

V današnjem času so pametne mobilne naprave vseprisotne. Ljudje jih imajo ves čas pri roki in si z njimi pomagajo pri vsakodnevnih opravilih. Med njimi je tudi načrtovanje poti, saj obstaja vrsta aplikacij, ki znajo napovedati trajanje potovanja do nekega cilja. Vendar pa gre pri tem za splošne napovedi, ki ne držijo za vsakogar, saj se tempo hoje ali vožnje med ljudmi razlikuje.

Predvsem poti, ki jih opravljamo pogosto, želimo optimizirati in s tem prihraniti čas. Izbira najprimernejšega prevoznega sredstva včasih ni očitna, nekatere krajše poti se na primer bolj splača opraviti s kolesom kot avtomobilom ali avtobusom. Tudi izbira poti do cilja ni vedno lahka, saj morda na najkrajši poti vedno naletimo na rdeč semafor. Iz tega izhaja ideja po prilagojeni analizi trajanja potovanj z različnimi prevoznimi sredstvi, s katero bi bilo lažje določiti najprimernejši čas odhoda. Pri tem lahko pomaga nabor senzorjev, ki ga vsebujejo sodobne mobilne naprave.

Namen diplomske naloge je razvoj mobilne aplikacije, ki omogoča razpoznavanje prevoznih sredstev in shranjevanje poti za kasnejšo analizo. Razločuje naj med hojo, kolesarjenjem, vožnjo z avtomobilom, vožnjo z mestnim avtobusom ter mirovanjem. Na ta način lahko uporabnik presodi, s katerim prevoznim sredstvom je prišel najhitreje do cilja ter na katerih poteh je bilo najmanj postankov.

Za dosego tega cilja so bile uporabljene tehnike strojnega učenja. V di-

plomskem delu je opisan postopek, ki je sestavljen iz treh delov. Prvi del obsega zbiranje senzoričnih podatkov različnih poti s pomočjo posebne mobilne aplikacije. Sledi njihova obdelava z algoritmi strojnega učenja, preko katerih so bili pridobljeni parametri za klasifikacijo. Ti so uporabljeni pri tretjem delu – izdelavi uporabniške aplikacije za razpoznavo prevoznih sredstev.

Poglavje 2

Teoretična podlaga

2.1 Nadzorovano učenje

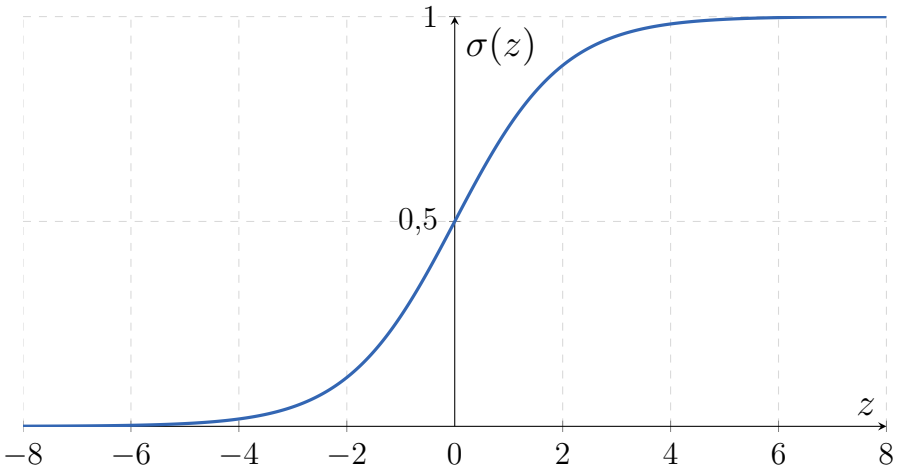
Nadzorovano učenje je oblika strojnega učenja, ki je sestavljeno iz učnega in izvajalnega dela. Učni algoritem iz množice vhodnih podatkov tvori novo znanje, ki ga imenujemo model ali hipoteza. Pri klasifikacijskih problemih je hipoteza diskretna funkcija, ki za vsak nov vhodni podatek x napove pripadajočo ciljno vrednost y [12, 13].

2.2 Logistična regresija

Kot obliko nadzorovanega učenja lahko smatramo regresijski model, statistično metodo za določanje relacij med podatki. Linearna regresija išče koeficiente linearne funkcije, ki določa relacijo med odvisno spremenljivko in atributi. Rezultat je hiperravnina, ki se čim bolj prilega vhodnim podatkom. Hipotezo $h_\theta(x)$ torej definiramo z enačbo (2.1), kjer je θ vektor parametrov in x vektor vhodnih vrednosti, pri čemer velja $x_0 = 1$ [12].

$$h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^\top x \tag{2.1}$$

Odvisna spremenljivka je zvezna in neomejena, zaradi česar linearna regresija ni primerna za klasifikacijo, kjer potrebujemo kategoričen izid. Pri



Slika 2.1: Sigmoidna funkcija.

binarni klasifikaciji odvisna spremenljivka zavzame eno izmed dveh možnosti $y \in \{0, 1\}$. Modelirati želimo pogojno verjetnostno porazdelitev $P(y = 1|x) = h_\theta(x)$, ki nam pove verjetnost, da ciljna spremenljivka spada v razred 1. Ker gre za Bernoullijevo porazdelitev, velja $P(y = 0|x) = 1 - P(y = 1|x)$. Problem rešimo z uvedbo vezne funkcije logit, ki predstavlja naravni logaritem obeta, da odvisna spremenljivka predstavlja eno izmed obeh kategorij (2.2).

$$\text{logit}(h_\theta(x)) = \log\left(\frac{h_\theta(x)}{1 - h_\theta(x)}\right) = \theta^\top x \tag{2.2}$$

Po preureditvi enačbe dobimo formulo za izračun hipoteze logistične regresije (2.3).

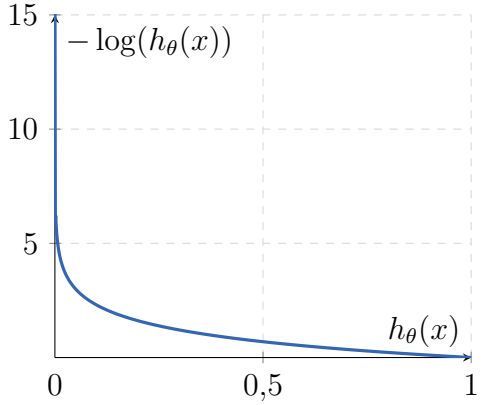
$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}} = \sigma(\theta^\top x) \tag{2.3}$$

Hipotezo predstavlja sigmoidna funkcija linearne kombinacije vhodnih vrednosti in uteži. Sigmoidna funkcija vrača le vrednosti na intervalu $(0, 1)$, pri čemer se pri višjih vhodnih vrednostih bliža vrednosti 1, pri nižjih pa vrednosti 0, kot prikazuje slika 2.1. Naučeni model logistične regresije napove, da primer spada v razred 1, če $h_\theta(x) \geq 0.5$, sicer spada v razred 0 [14, 16].

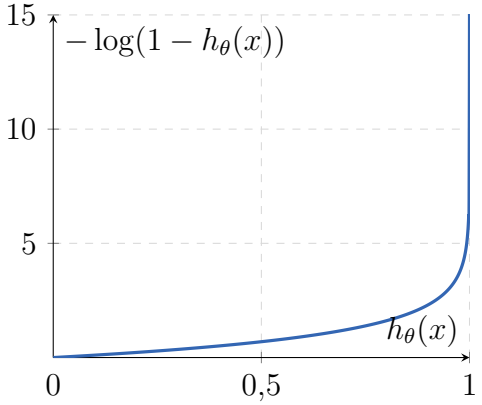
2.3 Optimizacijski problem

Vektor parametrov θ izberemo z minimizacijo funkcije napake. Ker je hipoteza pri logistični regresiji nelinearna, bi bila funkcija vsote kvadratov napake nekonveksna in s tem neprimerna za optimizacijo. Iz metode največjega verjetja izpeljemo konveksno funkcijo napake (2.4), ki predstavlja naš optimizacijski problem. Funkcija uporabi m testnih primerov (x, y), pri čemer x predstavlja vektor značilk, $y \in \{0, 1\}$ pa določa razred. Kot je razvidno iz slik 2.2 in 2.3, funkcija napake pri pravilni napovedi vrne vrednost 0, pri napačnih napovedih pa vrača vrednosti, ki se bližajo neskončnosti [14].

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right) \tag{2.4}$$



Slika 2.2: Funkcija napake pri $y = 1$ v odvisnosti od napovedi.



Slika 2.3: Funkcija napake pri $y = 0$ v odvisnosti od napovedi.

2.4 Gradientni spust

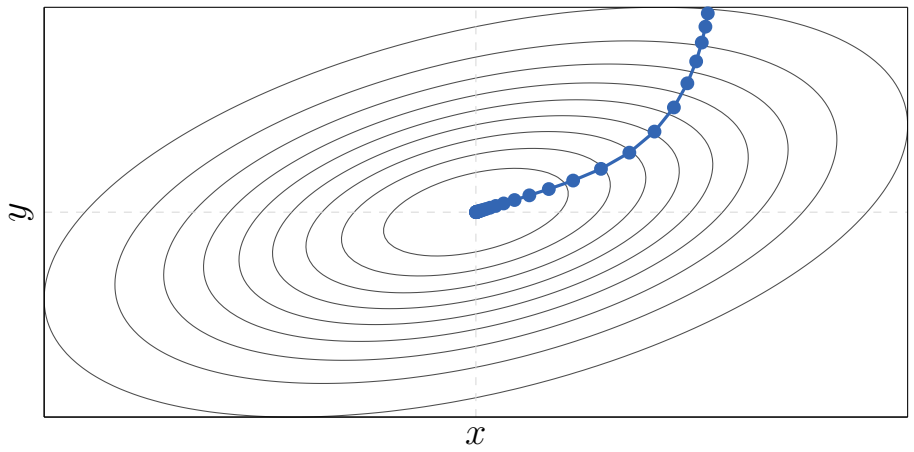
Gradientni spust je iterativna metoda za iskanje lokalnega minimuma funkcije. Po začetno poljubno izbranem vektorju $\theta^{(0)}$ se v vsakem koraku za faktor α premaknemo v nasprotni smeri gradienta funkcije napake $\nabla J(\theta^{(i)})$,

kot ponazarja slika 2.4. Pri tem faktorja ni potrebno manjšati, saj se korak bližje minimumu zaradi manjšega gradienta zmanjša, v samem minimumu pa se ustavi. Metoda tako ponavlja enačbo (2.5), dokler ne skonvergira.

$$\theta^{(i+1)} = \theta^{(i)} - \alpha \nabla J(\theta^{(i)}) \tag{2.5}$$

Enačba (2.6) prikazuje gradient funkcije napake pri logistični regresiji. Zaradi njene konveksnosti je gradientni spust primeren za iskanje globalnega minimuma [14].

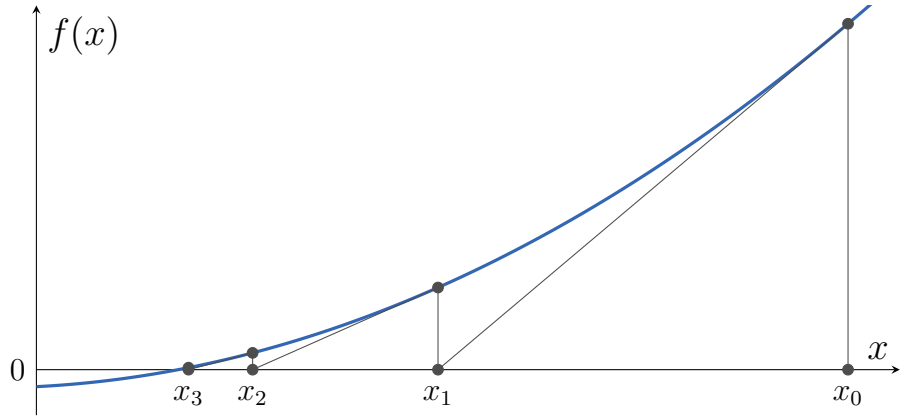
$$\nabla J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left((h_\theta(x^{(i)}) - y^{(i)}) x^{(i)} \right) \tag{2.6}$$



Slika 2.4: Prikaz delovanja gradientnega spusta na konturnem grafu konveksne funkcije.

2.5 Newtonova metoda

Newtonova metoda je iterativna metoda za iskanje ničel zvezno odvedljive funkcije. V vsakem koraku za dan približek ničle $x^{(i)}$ s pomočjo odvoda $f'(x)$ izračuna tangento in vrne njen presek z abscisno osjo $x^{(i+1)}$, ki predstavlja boljši približek ničle funkcije $f(x)$ (2.7). Postopek je ponazorjen na



Slika 2.5: Primer delovanja Newtonove metode.

sliki 2.5 [15] .

$$x^{(i+1)} = x^{(i)} - \frac{f(x^{(i)})}{f'(x^{(i)})} \tag{2.7}$$

Pri optimizaciji ji namesto same funkcije napake podamo njen odvod, saj se v ničlah odvoda nahajajo ekstremi funkcije. Če je funkcija napake dvojno odvedljiva in konveksna, Newtonova metoda poišče njen minimum. Iterativno formulo lahko posplošimo za več dimenzij z zamenjavo odvoda funkcije z njenim gradientom $\nabla J(\theta)$ in recipročne vrednosti dvojnega odvoda z inverzom Hessove matrike H^{-1} (2.8).

$$\theta^{(i+1)} = \theta^{(i)} - \mathrm{H}^{-1}\nabla J(\theta^{(i)}) \tag{2.8}$$

Hessova matrika je kvadratna matrika, sestavljena iz drugih parcialnih odvodov realne funkcije, ki jo definira enačba (2.9).

$$\mathrm{H}_{ij} = \frac{\partial^2 J}{\partial\theta_i \partial\theta_j} \tag{2.9}$$

Če uporabimo funkcijo napake za logistično regresijo, lahko Hessovo matriko izračunamo z enačbo (2.10).

$$\mathrm{H} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})(1 - h_\theta(x^{(i)}))(x^{(i)})(x^{(i)})^\top\right) \tag{2.10}$$

2.6 Regularizacija

Pri večjem številu značilk lahko pride do prekomernega prileganja učni množici, kar lahko rešimo z regularizacijo. Deluje tako, da parametrom doda uteži in s tem zmanjša vrednosti členov v hipotezi, kar jo zgladi. Funkciji napake prištejemo regularizacijski člen $R(\theta)$, sestavljen iz regularizacijskega parametra λ, ki določa moč regularizacije, in približka Evklidske norme (izpuščen je le θ_0), zaradi česar to obliko regularizacije imenujemo L_2 regularizacija, konstanti m in n pa predstavljata velikost učne množice oziroma dolžino vektorja parametrov (2.11) [14].

$$R(\theta) = \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 \tag{2.11}$$

Poglavje 3

Zbiranje podatkov

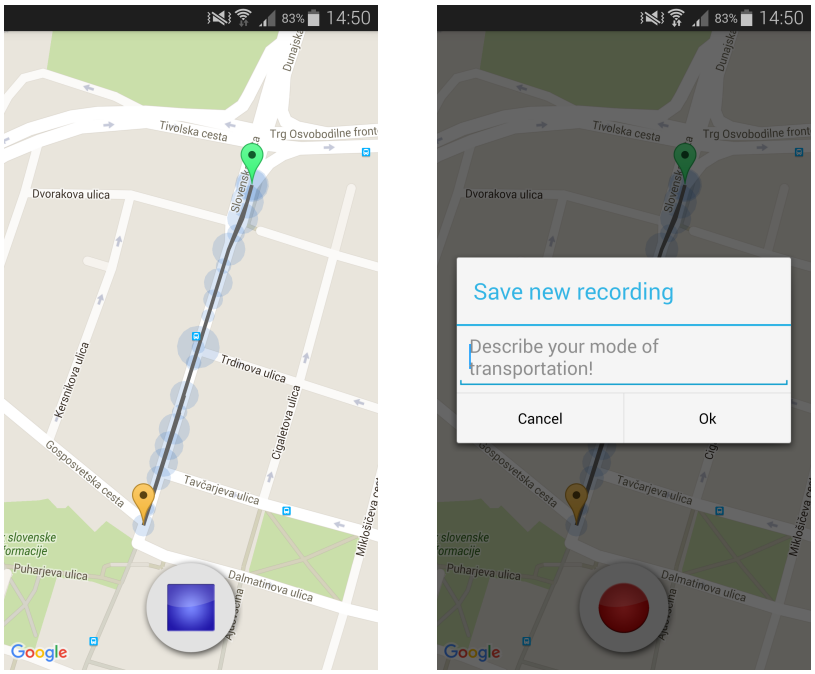
Razvita je bila mobilna aplikacija za zbiranje podatkov, ki so bili kasneje uporabljeni kot učna množica.

3.1 Uporabniški vmesnik

Android aplikacije uporabniški vmesnik prikazujejo v t. i. aktivnostih, ki predstavljajo celoten prikaz na zaslonu. Glavna aktivnost mobilne aplikacije za zbiranje podatkov je sestavljena iz gumba za snemanje poti in zemljevida. Za prikaz slednjega je bil uporabljen programski vmesnik Google Maps Android API, s katerim se zemljevid v aktivnost vključi znotraj fragmenta. Pri tem morajo biti na voljo storitve Google Play, kar je možno preveriti s pomožnim razredom `GoogleApiAvailability`. V kolikor so storitve na voljo, je možno do zemljevida dostopati preko vmesnika za komunikacijo s fragmenti `FragmentManager` in klica `getMap` na najdenem fragmentu.

Ob kliku na gumb za snemanje se prične izrisovati opravljena pot, kar omogočajo funkcije `GoogleMap` objekta. Za vsako lokacijo se izriše območje zaupanja, v katerem se uporabnik z 68% verjetnostjo nahaja (radij kroga je enak natančnosti lokacije GPS), celotna pot pa je sestavljena iz lomljenih črt med začetnim in končnim markerjem. Po koncu snemanja se prikaže dialog, v katerem lahko uporabnik opiše prevozna sredstva in druge podrobnosti

posnete poti, ki bi lahko kasneje pomagale pri ročni prepoznavi podatkov. Uporabo aplikacije ponazarjata posnetka zaslona na sliki 3.1.

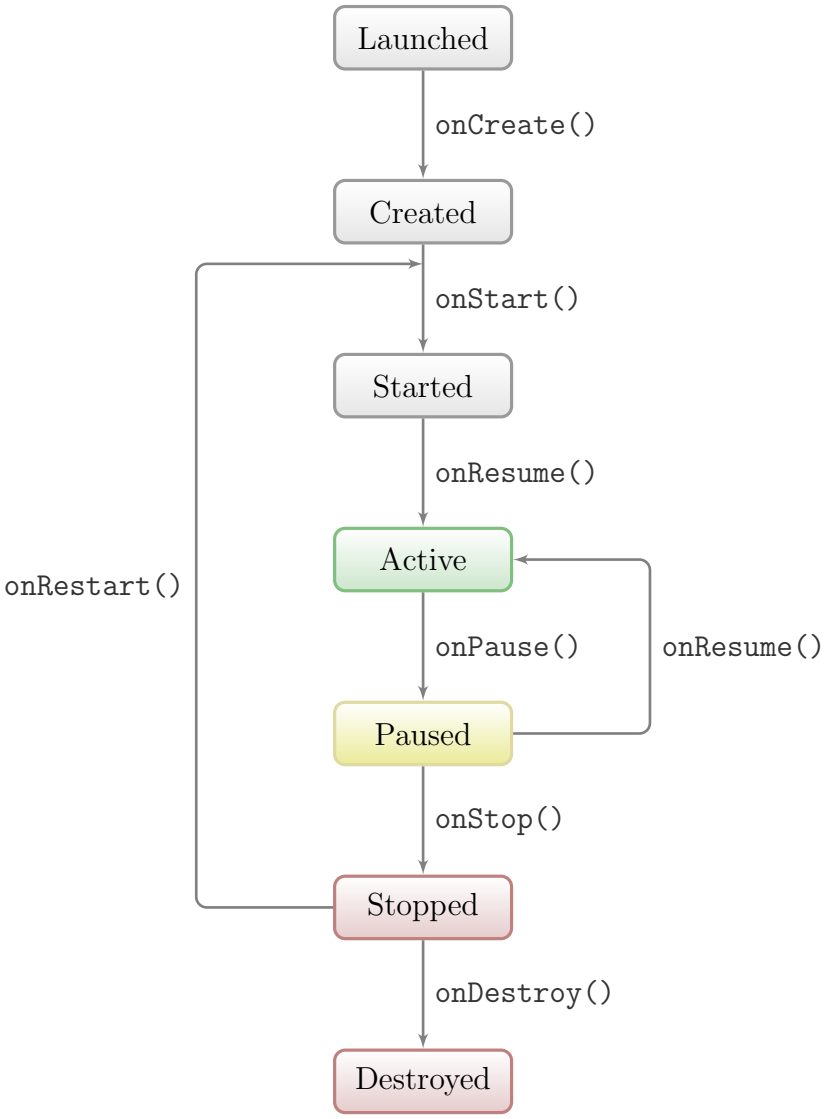


Slika 3.1: Primer prikaza poti med snemanjem in dialoga za opis prevoznega sredstva.

3.2 Storitev za zajem podatkov

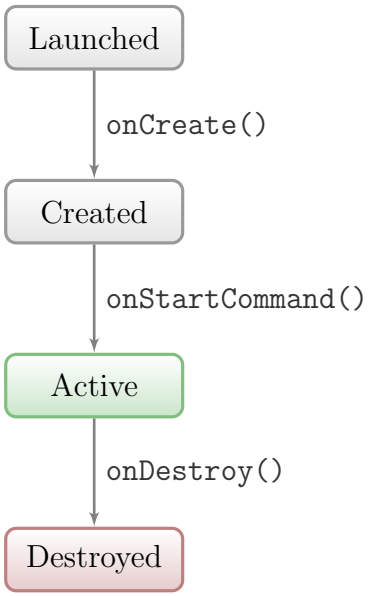
Gumb za snemanje sproži začetek zajema podatkov, ki poteka vse dokler uporabnik ne klikne na gumb za ustavitev. Če se vmes ugasne zaslon naprave, to ne sme vplivati na zbiranje podatkov. Poleg tega je zaželeno, da je tudi med snemanjem mogoče uporabljati mobilni telefon za klice in druge aplikacije. Če bi zajem podatkov potekal znotraj aktivnosti, to ne bi bilo mogoče. Omejen je namreč z življenjskim ciklom, ki ga prikazuje slika 3.2. Določa štiri statična stanja: aktivno (kadar je aktivnost v ospredju), prekinjeno (aktivnost je delno zakrita z drugo aktivnostjo), ustavljeno (aktivnost je v ozadju in ni več vidna) in uničeno stanje (aktivnost se ne izvaja več).

V ustavljenem stanju lahko operacijski sistem kadarkoli uniči aktivnost, da sprosti vire, zato zbiranje podatkov v ustavljeni aktivnosti ni zanesljivo [3].



Slika 3.2: Življenjski cikel aktivnosti.

Namesto tega se za dolgotrajne procese v ozadju uporablja storitev, aplikacijsko komponento brez uporabniškega vmesnika. Ker poteka ločeno od aktivnosti, ki jo je zagnala, lahko nadaljuje z izvajanjem, tudi ko je aktivnost uničena. Življenjski cikel je enostavnejši, saj nima prekinjenega in ustavljenega stanja [6]. Ponazarja ga slika 3.3.



Slika 3.3: Življenjski cikel storitve.

Storitev lahko komunicira z drugimi komponentami preko lokalnega oddajnika sporočil `LocalBroadcastManager`. Ko glavna aktivnost za zajem podatkov zažene storitev in registrira ustrezen prejemnik sporočil, lahko na zemljevid izrisuje pot. Storitev namreč pridobljene podatke sproti shranjuje v seznam in pošilja glavni aktivnosti. Sporočila, ki se lahko pošiljajo med komponentami aplikacije, morajo biti objekti razreda `Intent`, lokacijski in senzorični objekti pa se vanj vključijo s pomočjo serializacije. Ko uporabnik ustavi snemanje podatkov, glavna aktivnost ustavi storitev, ki pred svojim uničenjem aktivnosti vrne podatke celotne poti.

3.3 Lokacija

Globalni navigacijski sistemi delujejo na osnovi mreže satelitov, ki v enakomernih intervalih oddajajo informacije o svoji lokaciji in času. Ko sprejemnik prestreže sporočila vsaj štirih satelitov, izračuna njihovo oddaljenost glede na čas potovanja sporočil, nato pa z metodo trilateracije določi svoj položaj. Sodobna strojna oprema pametnih telefonov običajno vključuje nabor čipov,

ki omogoča uporabo več globalnih satelitskih sistemov. Ruski navigacijski sistem GLONASS je namreč za Evropo pogosto natančnejši od ameriškega sistema GPS, uporaba satelitov obeh sistemov skupaj pa omogoči zelo natančno določanje položaja [1].

Največja slabost pridobivanja lokacije s pomočjo satelitskih sistemov je potreba po neovirani poti do satelita, zaradi česar sistem v zaprtih prostorih ne deluje. Poleg tega lahko kot posledica nizke bitne hitrosti komunikacijskega kanala določitev začetne lege traja veliko časa, kar povzroča tudi visoko porabo energije. Zato je priporočen način določanja lokacije na pametnih mobilnih telefonih kombinacija satelitskih sistemov in informacij iz omrežja.

Grobo lokacijo je mogoče ugotoviti tudi iz podatkov mobilnega omrežja, saj se telefon nahaja v okolici baznih postaj, na katere se povezuje. Predvsem v urbanih okoljih pa je uspešno tudi ugotavljanje lokacije na osnovi moči zaznanih Wi-Fi dostopnih točk v bližini. Google preko Android uporabnikov konstantno zbira in posodablja zemljevid Wi-Fi dostopnih točk in pripadajočih koordinat, kar omogoča preprosto poizvedovanje lokacije, ki je hitrejše in v zaprtih prostorih tudi natančnejše od satelitskih sistemov pridobivanja lokacije [4].

V naši aplikaciji je bil uporabljen programski vmesnik `FusedLocationApi` iz sklopa Googlovih lokacijskih storitev, ki glede na podane parametre sam določa in kombinira vire za pridobitev lokacije. Za potrebe zbiranja podatkov je bila določena meja za natančnost lokacij 50 metrov. Inicializacija sistema za pridobitev lokacij je prikazana v izvorni kodi 3.3.1.

3.4 Pospešek

Pospeškomer je v sodobnih pametnih mobilnih napravah nepogrešljiv del strojne opreme, saj denimo omogoča zasuk zaslona glede na orientacijo naprave. Za razliko od ugotavljanja lokacije gre pri določanju pospeška za natančne in energijsko varčne meritve. V večini primerov je uporabljen ka-

```
@Override
public void onCreate() {
   super.onCreate();

   mLocationRequest = new LocationRequest();
   mLocationRequest.setPriority(
      LocationRequest.PRIORITY_HIGH_ACCURACY);
   mLocationRequest.setInterval(UPDATE_INTERVAL);
   mLocationRequest.setFastestInterval(FASTEST_INTERVAL);

   mGoogleApiClient = new GoogleApiClient.Builder(this)
      .addConnectionCallbacks(this)
      .addOnConnectionFailedListener(this)
      .addApi(LocationServices.API)
      .build();
   mGoogleApiClient.connect();
}

@Override
public void onConnected(Bundle dataBundle) {
   LocationServices.FusedLocationApi.requestLocationUpdates(
      mGoogleApiClient, mLocationRequest, this);
}
```

Izvorna koda 3.3.1: Vzpostavitev pridobivanja lokacije s pomočjo Googlovih lokacijskih storitev.

pacitiven MEMS (mikroelektromehanski sistem), ki meri linearni pospešek na treh oseh [9]. Ob tem je vštet tudi gravitacijski pospešek, tako da je magnituda pospeška naprave v mirujočem stanju okoli 9,81 m/s^2, v prostem padu pa 0 m/s^2.

Branje senzoričnih podatkov omogoča objekt `SensorManager`, s katerim se registrira vmesnik za poslušanje senzoričnih dogodkov `SensorEventListener`. Sistem glede na podano frekvenco vrača izmerjene vrednosti pospeška vseh treh osi. Ker pri določanju načina premikanja orien-

tacija naprave ne sme vplivati na rezultate, je bila uporabljena le magnituda pospeška.

3.5 Shranjevanje podatkov

Ker zajem podatkov poteka v storitvi, jih lahko med snemanjem hranimo v obliki seznamov v delovnem pomnilniku. Čeprav se konstantno povečujejo, so dovolj majhni, da bi tudi po eni uri snemanja zavzeli samo okoli 25 MB, običajno pa so poti bistveno krajše. Po končanem snemanju se podatki skupaj z opisom, ki ga uporabnik doda na koncu, zapišejo kot nov vnos v datoteko v eksternem pomnilniku naprave. V ta namen je uporabljen objekt `BufferedOutputStream`, ki so mu podani podatki v obliki bajtov.

Poglavje 4

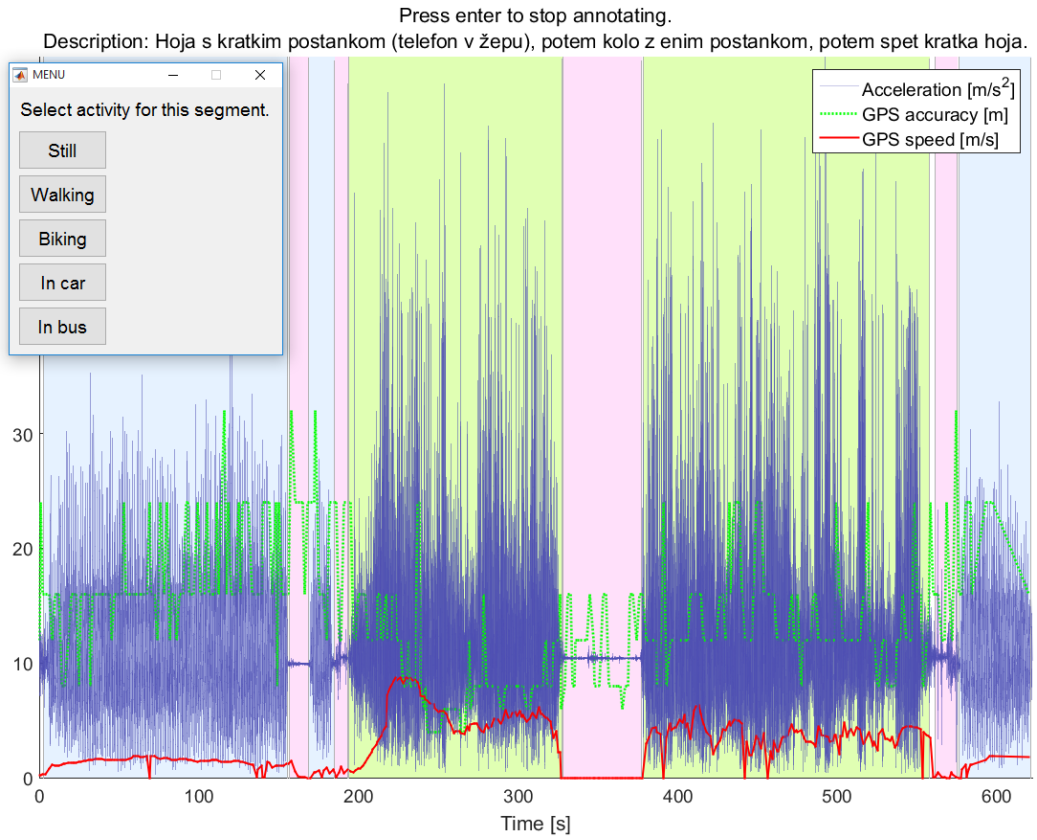
Učenje modelov

Obdelava zbranih podatkov je potekala s pomočjo programskega okolja MATLAB, naučeni modeli pa so bili kasneje uporabljeni za sprotno klasifikacijo.

4.1 Grafično razvrščanje podatkov

Zbrani podatki so bili ločeni po načinu premikanja in razdeljeni v pripadajoča polja struktur. V ta namen je bil razvit grafični vmesnik, ki omogoča ročno označevanje podatkov v časovnem grafu. Za posamezen posnetek so bili izrisani pospešek, hitrost in natančnost GPS v odvisnosti od časa, kar skupaj s pripadajočim opisom zadostuje za prepoznavo predelov aktivnosti. V redkih primerih, ko so bili podatki dvoumni, so bili le-ti izpuščeni iz učne množice. Prepoznana območja so bila določena s klikom na graf v točki začetnega in končnega časa, nakar se je odprl meni za izbiro pripadajoče aktivnosti, kot prikazuje slika 4.1. Za vsako aktivnost (mirujoče stanje, hoja, kolesarjenje, vožnja z avtomobilom in vožnja z avtobusom) je bilo ustvarjeno ločeno polje struktur, v katero so bili shranjeni označeni sklopi podatkov (zaporedne vrednosti hitrosti, GPS natančnosti in pospeška s časovnimi oznakami).

Mirujoče stanje je najlažje prepoznati, kadar je hitrost enaka 0 m/s in pospešek približno 9,81 m/s^2 (gravitacijski pospešek), vendar je potrebno

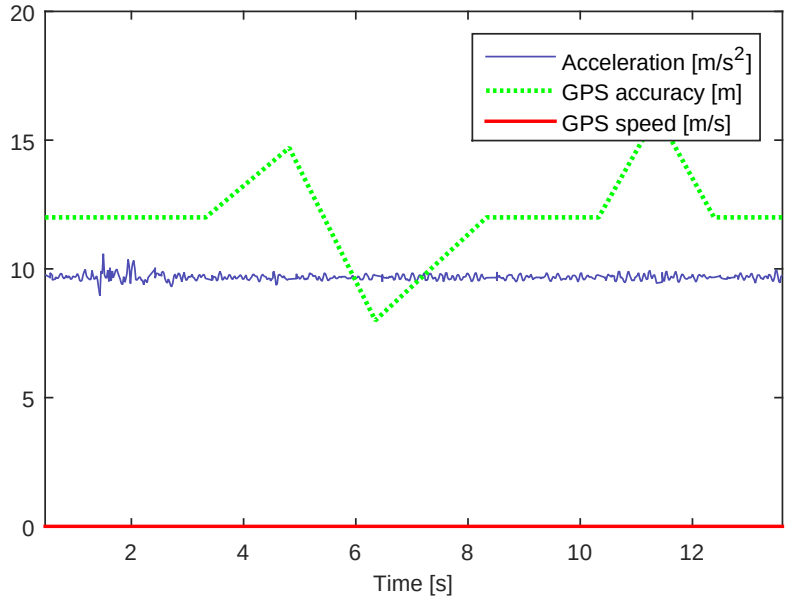


Slika 4.1: Primer označevanja podatkov. Določena so območja hoje (modra), mirovanja (rdeča) in kolesarjenja (zelena).

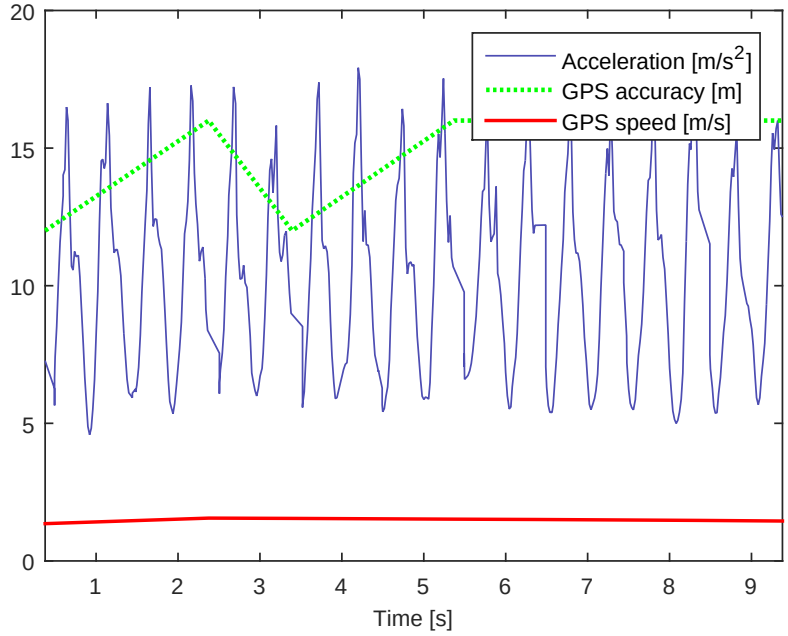
upoštevati tudi manjša nihanja v pospešku (zaklepanje kolesa, prestopanje med čakanjem ipd.) in hitrosti, kadar je slabša natančnost GPS (npr. v zaprtih prostorih). Primer grafa mirovanja je prikazan na sliki 4.2.

Za hojo je značilen ponavljajoč se vzorec nihanja pospeška v odvisnosti od časa, v katerem je možno prepoznati korake. Če se mobilna naprava nahaja v hlačnem žepu, ima vsak drugi korak nekoliko večji pospešek, sicer pa so koraki približno enakomerni, kot je razvidno s slike 4.3. Standardni odklon pospeška se giblje med 2 in 3 m/s^2, hitrost hoje pa je večino časa konstantna in pri hitri hoji dosega vrednosti do 2.5 m/s.

Med kolesarjenjem prihaja do močnejših tresljajev (robniki, tlakovci, slaba cesta), zaradi česar je pospešek zelo variabilen. Maksimalne vrednosti

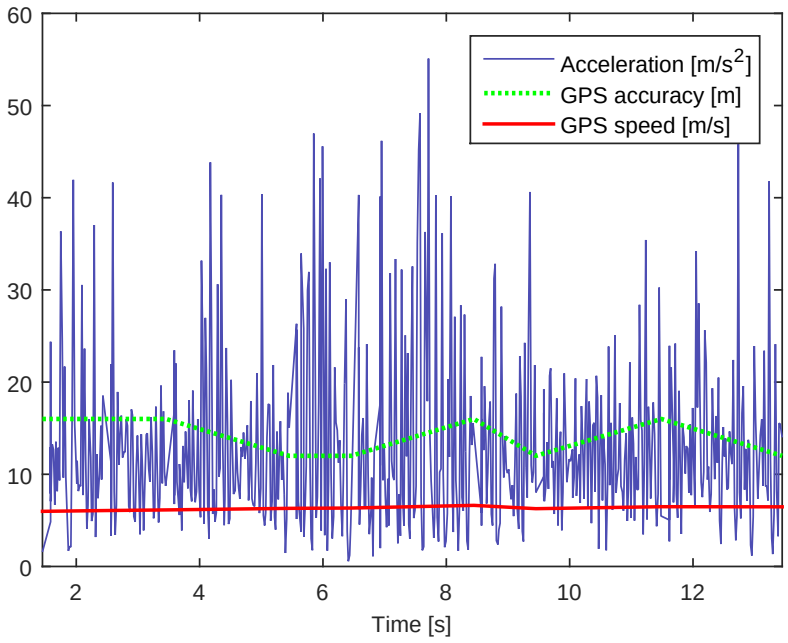


Slika 4.2: Primer grafa mirujočega stanja.



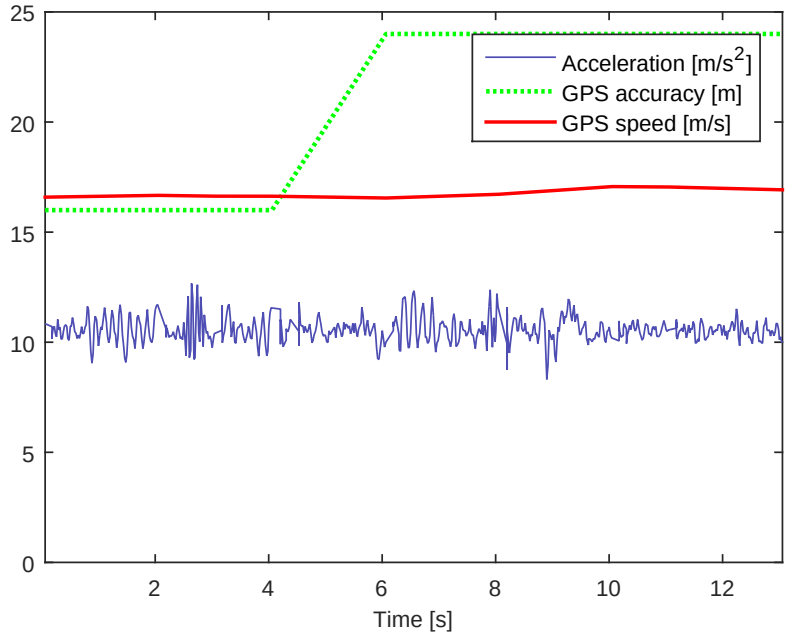
Slika 4.3: Primer grafa hoje.

se gibljejo okoli 70 m/s^2, standardni odklon posameznega segmenta pa do 7 m/s^2. Tudi hitrost se za razliko od hoje pogosteje spreminja, kar je posledica prilagajanja okoliščinam (hitrejša vožnja na kolesarski stezi, počasnejša v peš coni), dosega pa vrednosti do 6 m/s. Primer grafa kolesarjenja je prikazan na sliki 4.4.

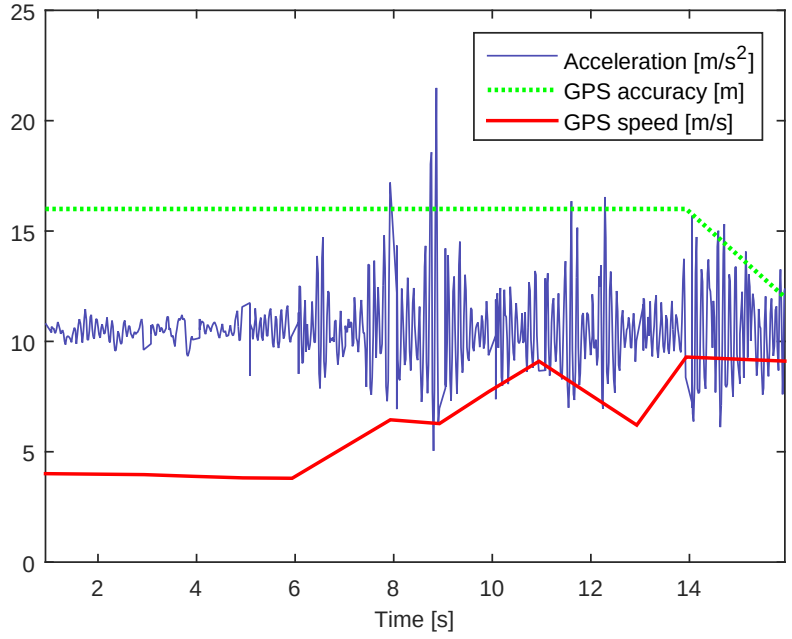


Slika 4.4: Primer grafa kolesarjenja.

Za vožnjo z avtomobilom so značilni stalni rahli tresljaji, tako da v večini primerov standardni odklon pospeška znaša manj kot 1 m/s^2. Dosegajo se daleč najvišje hitrosti (do 36 m/s na avtocesti oz. okoli 16 m/s v mestu, kot prikazuje slika 4.5). Zaradi avtomobilske strehe je lahko GPS manj natančen.



Slika 4.5: Primer grafa vožnje z avtomobilom.



Slika 4.6: Primer grafa vožnje z mestnim avtobusom.

Vožnja z mestnim avtobusom je v osnovi podobna vožnji z avtomobilom, vendar ima nekaj pomembnih razlik: pospešek obsega nekoliko širši razpon vrednosti (standardni odklon v povprečju znaša do 1.5 m/s^2), dosega nižje hitrosti (maksimalno 15 m/s) in pojavlja se v krajših segmentih, saj ima posamezna pot veliko več postankov kot pri vožnji z avtomobilom. Tudi tu zaradi strehe prihaja do občasnih slabših meritev GPS. Primer grafa vožnje z avtobusom je prikazan na sliki 4.6.

4.2 Pridobivanje značilk

Po označbi vseh posnetkov so bile iz zgrajenih polj struktur pridobljene značilke. Za vsako strukturo je bila uporabljena tehnika drsečega okna dolžine 4 sekunde s 50% prekrivanjem, tj. zaporeden pomik po podatkih, ki se nahajajo znotraj začetnega in končnega časa, določenega z oknom. V vsakem koraku so bile iz zaobjetih podatkov izračunane sledeče značilke:

- povprečni pospešek,
- povprečna hitrost,
- standardni odklon pospeška,
- standardni odklon hitrosti,
- maksimalna hitrost,
- povprečje natančnosti signala GPS,
- maksimalna vrednost natančnosti signala GPS,
- povprečna moč pospeška,
- povprečna moč hitrosti,
- spektralna entropija pospeška,
- spektralna entropija hitrosti.

Velja pripomniti, da vrednost natančnosti GPS pomeni radij območja, v katerem se uporabnik verjetno nahaja, tako da višja vrednost nakazuje na slabšo natančnost. Pri izračunu značilk je potrebno zagotoviti, da so vse vrednosti normalizirane oz. neodvisne od dolžine podatkov, ker imajo pametne mobilne naprave različne frekvence vzorčenja. Povprečna moč je bila izračunana po formuli (4.1), kjer $x(t_i)$ predstavlja i-to vrednost v n vzorcih [2].

$$E = \frac{1}{n} \sum_{i=0}^{n} x(t_i)^2 \tag{4.1}$$

Spektralna entropija je bila izračunana po formuli (4.2), kjer $P(f)$ (4.3) predstavlja gostoto verjetnosti, ki je sestavljena iz normalizirane spektralne gostote moči, pri čemer je $\hat{x}(f)$ Fourierova transformacija vhodnega signala $x(t)$ [11].

$$H = -\sum_{i=0}^{n} P(f_i) \log P(f_i) \tag{4.2}$$

$$P(f) = \frac{|\hat{x}(f)|^2}{\sum_{i=0}^{n} |\hat{x}(f_i)|^2} \tag{4.3}$$

Izračunane značilke so bile shranjene v matriki značilk pripadajoče aktivnosti. Kadar v drsečem oknu ni bilo podatkov GPS, so bile izračunane samo značilke na osnovi pospeška. Te so se shranile v ločeno matriko brez podatkov GPS.

4.3 Predpriprava učne množice

Za uporabo binarne logistične regresije na več razredih se klasifikacijski problem razbije na več dvorazrednih problemov [12]. Ustvariti je torej potrebno model za vsak način premikanja. V našem primeru sta bila ustvarjena po dva modela za vsak razred oz. aktivnost – za klasifikacijo primerov brez in s podatki GPS.

Za učenje modelov z logistično regresijo je bilo potrebno za vsak model zgraditi matriko vhodnih podatkov X in vektor izhodnih vrednosti y. Vsaka vrstica matrike X je sestavljena iz skupine značilk, ki predstavljajo posamezen učni primer iz matrik značilk vseh aktivnosti v izbrani GPS kategoriji. Vektor y pa za vsak učni primer določi pripadnost k razredu modela (privzame vrednost 1 pri primerih pripadajočega razreda ter vrednost 0 pri vseh drugih razredih).

4.3.1 Normalizacija značilk

K hitri optimizaciji pripomore, da značilke zavzemajo približno enak obseg vrednosti. Gradientni spust ima namreč težave s konvergiranjem pri različno velikih značilkah, kjer so gradienti usmerjeni skoraj pravokotno na najkrajšo pot do minimuma in pride do skakanja.

Za normalizacijo je bila uporabljena standardna vrednost značilk, izračunana po enačbi (4.4), kjer μ predstavlja povprečje, σ pa standardni odklon skupine značilk, v katere spada x. Na ta način se srednja vrednost vseh skupin značilk giblje okoli 0, varianca pa je enaka 1.

$$z = \frac{x - \mu}{\sigma} \tag{4.4}$$

4.3.2 Preslikavanje značilk

Linearna kombinacija značilk in parametrov logistične regresije, ki je uporabljena v hipotezi, določi linearno mejo med razredoma, kar ne zagotavlja najboljšega prileganja podatkom. Meja med razredoma je namreč pogosto polinom višje stopnje. Rešitev je vpeljava novih značilk v obliki vseh možnih kombinacij zmnožkov danih značilk do določenega velikostnega reda. Enačba (4.5) prikazuje primer hipoteze z dvema značilkama, ki sta bili preslikani v polinom druge stopnje.

$$h_\theta(x) = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_2^2) \tag{4.5}$$

4.4 Minimizacija funkcije napake

Na predpripravljenih podatkih so bili izračunani parametri za logistično regresijo. Implementirana sta bila dva načina za minimizacijo funkcije napake: gradientni spust in Newtonova metoda. Za primerjavo so bili parametri izračunani tudi z vgrajenima MATLAB funkcijama `fminsearch` in `glmfit`. Funkcija `fminsearch` z uporabo iterativne metode brez računanja odvodov poišče lokalni minimum v bližini podane točke, funkcija `glmfit` pa vrača ocene parametrov posplošenih linearnih modelov, med katere spada tudi logistična regresija. Funkcija je bila uporabljena s parametroma `binomial` za porazdelitev in `logit` za povezovalno funkcijo.

```
function theta = gradient_descent_regularized(X, y, theta, ...
   alpha, n, precision, lambda)
   i = 0; eps = 100;
   cost = logreg_cost_regularized(X, y, theta, lambda);

   while eps > precision && i < n
      theta_old = theta;
      cost_old = cost;

      theta = theta_old - alpha * ...
         logreg_grad_regularized(X, y, theta_old, lambda);
      cost = logreg_cost_regularized(X, y, theta, lambda);

      eps = abs(cost_old - cost);
      i = i + 1;
   end
end
```

Izvorna koda 4.4.1: Implementacija regulariziranega gradientnega spusta.

Implementacijo regulariziranega gradientnega spusta prikazuje izvorna koda 4.4.1. Funkciji so bili kot argumenti podani predpripravljena matrika značilk `X`, vektor izhodnih podatkov `y`, začetni približek vektorja parame-

```
function J = logreg_cost_regularized(X, y, theta, lambda)
   m = length(y);
   h = sigmoid(X * theta);
   J = (1/m) * (-y' * log(h) - (1 - y') * log(1 - h)) + ...
      (lambda/(2*m)) * sum(theta(2:end).^2);
end
```

Izvorna koda 4.4.2: Regularizirana funkcija napake.

```
function grad = logreg_grad_regularized(X, y, theta, lambda)
   m = length(y);
   grad = (1/m) * X' * (sigmoid(X * theta) - y) + ...
      (lambda/m)*[0; theta(2:end)];
end
```

Izvorna koda 4.4.3: Gradient regularizirane funkcije napake.

trov `theta`, faktor hitrosti učenja `alpha`, maksimalno število iteracij `n`, natančnost ocene parametrov `precision` in regularizacijski parameter `lambda`. Algoritem po formuli (2.5) iterativno izboljšuje oceno parametrov, dokler ne skonvergira do definirane natančnosti ali doseže maksimalno število iteracij. Funkciji `logreg_cost_regularized` in `logreg_grad_regularized`, ki za podane argumente izračunata napako oz. gradient, sta prikazani v izvornih kodah 4.4.2 in 4.4.3.

Implementacijo regularizirane Newtonove metode prikazuje izvorna koda 4.4.4. Funkciji so podani enaki argumenti kot pri gradientnem spustu (razen parametra za faktor hitrosti učenja, ki pri Newtonovi metodi ni potreben), kliče enaki funkciji za izračun gradienta in napake ter uporablja enak kriterij konvergence. Novo oceno parametrov v vsaki iteraciji izračuna po formuli (2.8), pri čemer je pri računanju Hessove matrike pomembna uporaba razpršenih matrik. Množiti je treba namreč diagonalne matrike, ki sicer zavzamejo skoraj kvadratno več prostora.

```
function theta = newtons_method_regularized(X, y, theta, ...
   n, precision, lambda)
   i = 0; eps = 100; m = length(y);
   cost = logreg_cost_regularized(X, y, theta, lambda);

   while eps > precision && i < n
      theta_old = theta;
      cost_old = cost;

      reg = (lambda/m) * eye(size(X, 2)); reg(1) = 0;
      H = (1/m) * (X' * spdiags(sigmoid(X*theta), 0, m, m) ...
         * spdiags(1 - sigmoid(X*theta), 0, m, m) * X) + reg;
      theta = theta - H\logreg_grad_regularized(X, y, theta, lambda);
      cost = logreg_cost_regularized(X, y, theta, lambda);

      eps = abs(cost_old - cost);
      i = i+1;
   end
end
```

Izvorna koda 4.4.4: Implementacija regularizirane Newtonove metode.

4.5 Validacija modelov

Za validacijo naučenih modelov je bilo uporabljeno k-kratno prečno preverjanje, s pomočjo katerega je bilo moč izbrati najbolj ustrezne argumente za optimizacijske algoritme. Deluje tako, da učne podatke s pomočjo vgrajene MATLAB metode `crossvalind` naključno razdeli na k delov. Učni algoritem se izvede k-krat, pri čemer se vsak del enkrat uporabi kot testna množica, preostalih $k - 1$ delov pa kot učna množica. V vsaki ponovitvi učenja so pridobljeni štirje vektorji parametrov za logistično regresijo (po en vektor za vsak način učenja), ki so uporabljeni za klasifikacijo testne množice. Za primerjavo rezultatov s pričakovanim izidom je bila uporabljena povprečna absolutna napaka, izračunana po enačbi (4.6), kjer je n število primerov v

testni množici, y vektor pričakovanih vrednosti, h pa vektor napovedanih vrednosti. Po k ponovitvah se izračuna še povprečje vrnjenih napak.

$$e = \frac{1}{n} \sum_{i=1}^{n} |h_i - y_i| \tag{4.6}$$

4.6 Shranjevanje rezultatov

Po vsakem učenju in validaciji modelov so bili rezultati shranjeni v namenski razred, ki vsebuje sledeče lastnosti: ime aktivnosti, ki jo model klasificira, red prečnega preverjanja, število ponovitev učenja modelov, normaliziranost podatkov, stopnja, do katere so bile značilke preslikane, parameter za regularizacijo, faktor hitrosti učenja, ki je bil uporabljen za gradientni spust, natančnost ocene parametrov pri gradientnem spustu in pri Newtonovi metodi ter povprečne napake za vse štiri načine izračuna parametrov (vgrajeni funkciji `fminsearch` in `glmfit` ter gradientni spust in Newtonova metoda). Na ta način je bilo možno zbrati in primerjati različno naučene modele in določiti najprimernejše argumente.

Poglavje 5

Sprotna klasifikacija

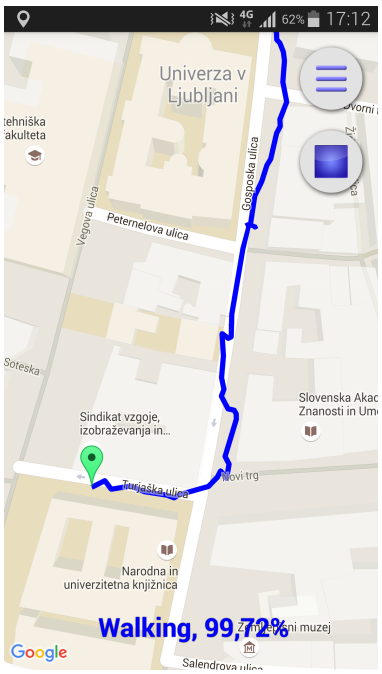
Za uporabo pridobljenih modelov je bila razvita mobilna aplikacija, ki omogoča sprotno klasifikacijo, shranjevanje poti s pripadajočimi načini premikanja in možnost naknadne korekcije klasificiranih poti.

5.1 Uporabniški vmesnik

Podobno kot pri aplikaciji za zbiranje podatkov glavno aktivnost sestavlja zemljevid, prikazan preko uporabniškega vmesnika Google Maps Android API. Gumb za snemanje zažene pridobivanje lokacijskih in senzoričnih podatkov ter klasifikacijsko logiko, ki na zemljevid sproti izpisuje način premikanja in pripadajočo verjetnost, opravljena pot pa se izrisuje v barvi ustrezne aktivnosti, kot prikazuje slika 5.1.

Aplikacija, ki pridobi senzorične podatke in sama razpozna način premikanja, je lahko uporabna za analizo in primerjavo porabe časa potovanj. Zato se posnetki shranjujejo za kasnejši ogled. Ob kliku na gumb za meni se prikaže možnost izbire med seznamom preteklih posnetkov in shranjenih lokacij. Slednji omogoča spreminjanje, brisanje in dodajanje pogosto obiskanih lokacij, kot prikazuje slika 5.2. Določitev pogosto obiskanih lokacij pomaga pri preglednejši ureditvi posnetkov.

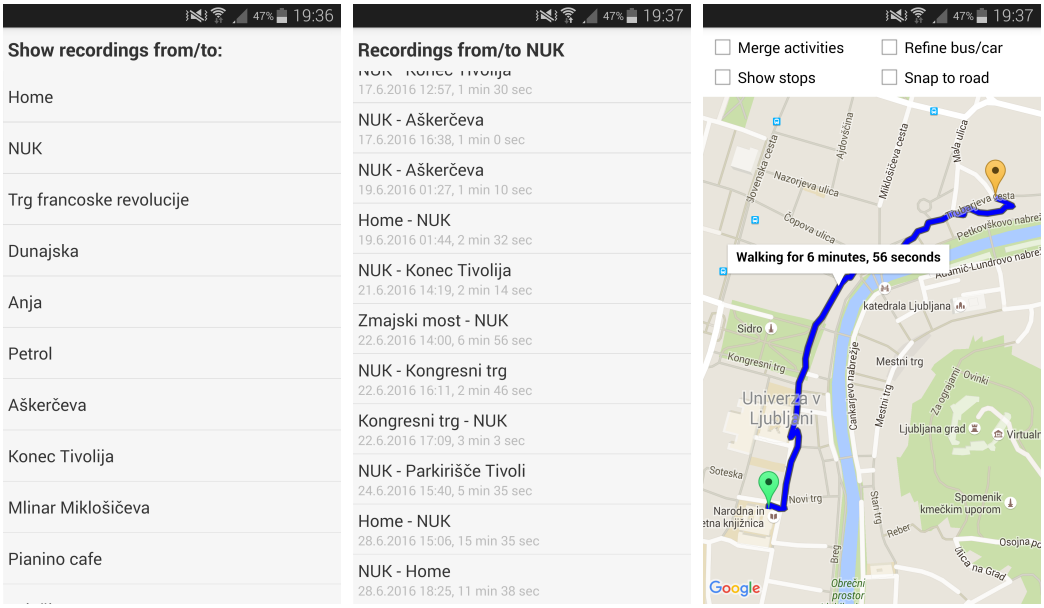
Seznam lokacij, ki so bile začetna ali končna točka posnetih poti, je do-



Slika 5.1: Primer sprotnega prikaza hoje v aplikaciji za klasifikacijo.

Slika 5.2: Prikaz seznama lokacij, ki ga lahko uporabnik ureja.

segljiv kot druga možnost v meniju. Ob kliku na eno izmed njih se izpiše seznam vseh pripadajočih posnetkov. Podroben ogled shranjenega posnetka je mogoč po kliku na izbran element v seznamu. Prikaže se opravljena pot, ki je po delih obarvana glede na način premikanja, kar je doseženo z izrisom več ločenih lomljenih črt. Klik na posamezen del izpiše način premikanja in porabljen čas v obliki informacijskega okna, ki ga omogočajo funkcije objekta `Marker`. Uporabnik ima na voljo tudi opciji za lepši prikaz posnetka (poravnava poti s cestami ter označbe postankov) ter možnosti za naknadno korekcijo klasifikacije (združitev osamelcev s sosednjimi segmenti ter razločitev med avtomobilom in avtobusom glede na postajališča).



Slika 5.3: Primer postopka za prikaz posnete poti. Po izbiri lokacije se odpre seznam ustreznih posnetkov, ki se po kliku izrišejo na zemljevid.

5.2 Storitev za klasifikacijo

Pridobivanje lokacije in pospeška v storitvi deluje na enak način kot v aplikaciji za zbiranje podatkov. Ko se nabere dovolj podatkov za dolžino časovnega

okna, se slednji za polovico pomakne naprej, zaobjeti podatki pa se uporabijo za klasifikacijo.

Logika za klasifikacijo deluje na osnovi statično določenih parametrov za logistično regresijo, pridobljenih med učenjem modelov. Uporablja deset vektorskih parametrov – po enega za model mirovanja, hoje, kolesarjenja, vožnje z avtomobilom in mestnim avtobusom ter enake modele brez podatkov GPS. Logika najprej izračuna značilke, naštete v poglavju 4.2, pri čemer v oknih brez lokacijskih podatkov izračuna samo značilke na osnovi pospeška. Sledi predpriprava učne množice, kjer morajo biti parametri za normalizacijo in preslikavanje značilk skladni s tistimi, ki so bili uporabljeni pri učenju modelov. V zadnjem koraku se s pomočjo parametrov θ po enačbi hipoteze logistične regresije (2.3) izračuna verjetnost za vsak model. Rezultat klasifikacije predstavlja način premikanja, ki ga določa model z najvišjo verjetnostjo. Storitev glavni aktivnosti odda sporočilo, ki vsebuje verjetnost in ime razpoznanega načina premikanja.

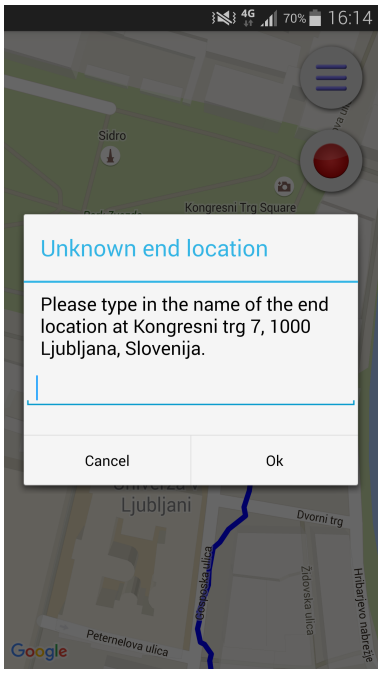
5.3 Shranjevanje in prikaz klasificiranih posnetkov

Android za shrambo strukturiranih podatkov nudi podporo za uporabo lokalne SQLite podatkovne baze, ki je dostopna vsem razredom aplikacije. Pomožni razred `SQLiteOpenHelper` omogoča ustvarjanje nove baze, pridobitev kurzorja za poizvedbe ter `SQLiteDatabase` objekta, na katerem je mogoče vršiti SQL stavke [8]. V naši aplikaciji sta bili ustvarjeni dve tabeli – za lokacije in posnetke. Prva vključuje ime lokacije, naslov, zemljepisno širino in dolžino, druga pa začetni čas posnetka, trajanje, seznam koordinat, časa in pripadajočih aktivnosti ter začetno in končno lokacijo. Slednja sta tuja ključa, ki referencirata vnosa iz tabele lokacij.

Med snemanjem se sklopi koordinat s skupnim načinom premikanja shranjujejo v namenskih objektih, ki vključujejo čas in izgled lomljene črte za ponoven izris na zemljevid. Po zaključku snemanja se seznam vseh ustvar-

jenih objektov v bazo zapiše preko serializacije v JSON niz, kar omogoča Googlova knjižnica Gson.

Pred zapisom v bazo se izvrši še dodelitev lokacij začetni in končni točki poti. V tabeli lokacij kazalec prehodi vse vnose in s pomočjo funkcij razreda `Location` računa razdalje med shranjenimi lokacijami in točkama. Za posamezno točko izbere najbližjo shranjeno lokacijo in če je razdalja med njima manjša od mejne vrednosti 150 metrov, posnetku za začetno oz. končno lokacijo poda ključ ustreznega vnosa v tabeli lokacij. Sicer pa se prikaže dialog, ki uporabnika poziva, naj določi ime za novo lokacijo, kot prikazuje slika 5.4. Pri tem mu navedemo tudi naslov, ki ga pridobimo s pomočjo t. i. obratnega geokodiranja. Gre za proces pridobivanja naslova iz koordinat, ki ga v Androidu omogoča objekt `Geocoder`. Koordinate, naslov in ime lokacije se shranijo kot nov vnos v tabeli lokacij, nato pa se posodobi še ustrezno polje trenutnega posnetka.



Slika 5.4: Primer prikaza dialoga po posnetku z novo končno lokacijo.

Uporabnik lahko lokacije ureja tudi v seznamu lokacij, dostopnem iz me-

nija. Posodabljanje, brisanje in ustvarjanje novih vnosov v tabeli lokacij omogočajo funkcije objekta `SQLiteDatabase`. V primeru sprememb se ponovno izvrši dodeljevanje lokacij posnetkom, s čimer je zagotovljeno, da imajo posnetki vedno dodeljene najbližje shranjene lokacije.

Za prikaz posnetkov se uporabniku najprej prikaže seznam shranjenih lokacij, s klikom na eno izmed njih pa se izvrši SQL poizvedba, ki izvede zunanji stik med tabelama lokacij in posnetkov po obeh poljih z zunanjim ključem in pogojem, da je vsaj eno izmed obeh polj enako izbrani lokaciji, kot prikazuje primer 5.3.1. Kazalec zapolni seznam preko vmesnika `SimpleCursorAdapter`, ki so mu podani ustrezni stolpci in pripadajoči objekti uporabniškega vmesnika.

```
SELECT startLoc.name AS startLocName,
       startLoc.address AS startLocAddress,
       endLoc.name AS endLocName,
       endLoc.address AS endLocAdress,
       recordings.* FROM recordings AS recordings
LEFT JOIN locations AS startLoc ON startLoc._id = recordings.startlocation
LEFT JOIN locations AS endLoc ON endLoc._id = recordings.endlocation
WHERE startLoc.name = 'Home' OR endLoc.name = 'Home';
```

Izvorna koda 5.3.1: Primer SQL poizvedbe za vse posnetke od in do doma.

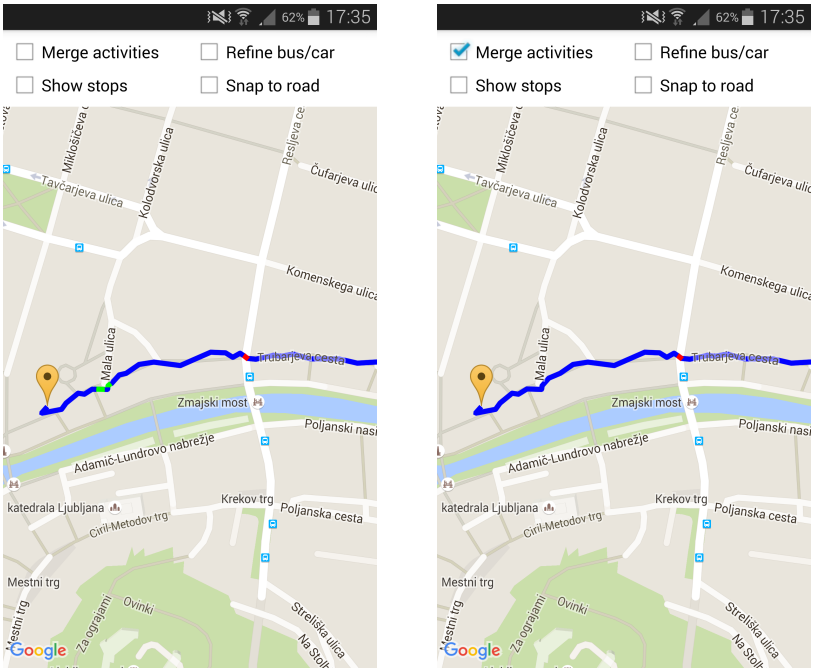
5.4 Naknadna obdelava posnetkov

Ob prikazu shranjenega posnetka je uporabniku na voljo več opcij za izris oz. korekcijo podatkov.

5.4.1 Združevanje osamelcev

Med klasifikacijo se občasno pojavijo krajši napačno klasificirani segmenti, ki v podatkih predstavljajo osamelce. Problem je rešen z algoritmom, ki poišče

sklope z enako aktivnostjo (z izjemo mirovanja), ki so krajši od določene meje 15 sekund. Če je bil sklop pred in po tem klasificiran kot mirovanje, se tudi ta sklop združi vanju, ker obstaja verjetnost, da je prišlo do napačne klasifikacije zaradi močnejšega premikanja mobilne naprave ali šumnih podatkov GPS. Sicer pa se segment združi v naslednjega ali prejšnjega, pri čemer ima prednost daljši segment, če gre pri obeh za nemirujočo aktivnost. Pri prehodu med mirovanjem in premikanjem pa lahko zaradi neznačilnih podatkov pride do osamelcev, ki večinoma predstavljajo ustavljanje oz. nadaljevanje poti, zato se združijo v sosednji segment, ki ni mirujoč. Primer združevanja osamelcev je prikazan na sliki 5.5.



Slika 5.5: Primer združevanja osamelcev pri posnetku hoje (modra) z vmesnim postankom (rdeča). Zelena predela (kolesarjenje) sta bila napačno klasificirana.

5.4.2 Poravnava poti s cestami

Ob slabšem signalu GPS je izrisana pot lahko nenatančna, zato je uporabniku ponujena opcija za poravnavo shranjenih poti s cestami. Omogoča jo vmesnik

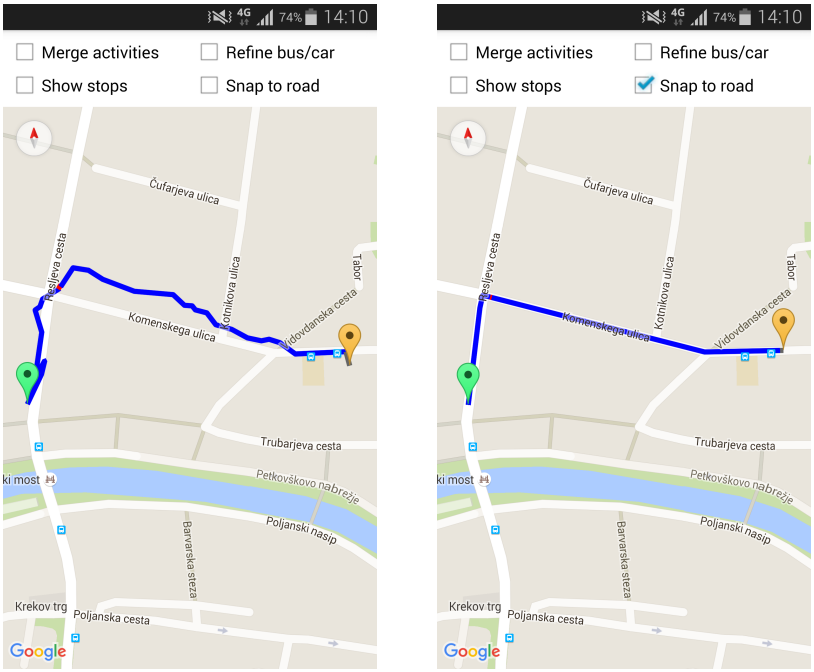
Google Maps Roads API preko metode `snapToRoads`, ki sprejme maksimalno 100 koordinat in vrne njihove ekvivalente, prilegajoče se cestam [7]. Komunikacija z vmesnikom deluje preko zahtevkov HTTP GET, ki jih omogoča Googlova odprtokodna knjižnica Volley. Zahtevku je podan URL v obliki `https://roads.googleapis.com/v1/snapToRoads?path=PATH&key=API_KEY`, pri čemer parameter `PATH` sestavljajo zaporedne koordinate, ločene z znakom `|`, parameter `API_KEY` pa predstavlja ključ, s katerim se aplikacija identificira. Ker je uporaba storitve omejena, se mora vsaka aplikacija v Googlovi konzoli za razvijalce registrirati in pridobiti svoj edinstven ključ.

Izvorna koda 5.4.1 prikazuje primer prejetega odgovora. Gre za seznam JSON koordinat s pripadajočimi indeksi originalnih točk.

```
{"snappedPoints": [
    {
      "location": {
        "latitude": 46.0514291218991,
        "longitude": 14.516596907842686
      },
      "originalIndex": 0,
      "placeId": "ChIJW2rf4H0tZUcRwPbLqlUQDW0"
    },
    ...
]}
```

Izvorna koda 5.4.1: Primer odgovora metode `snapToRoads`.

Posneta pot se razdeli na dele po maksimalno 100 točk, nakar se za vsak del pokliče metoda `snapToRoads`. Prejeti odgovori se po vrsti združijo in s pomočjo originalnih indeksov se zgradijo nove lomljene črte, ki jih sestavljajo prejete točke in originalne lastnosti, kot prikazuje slika 5.6.

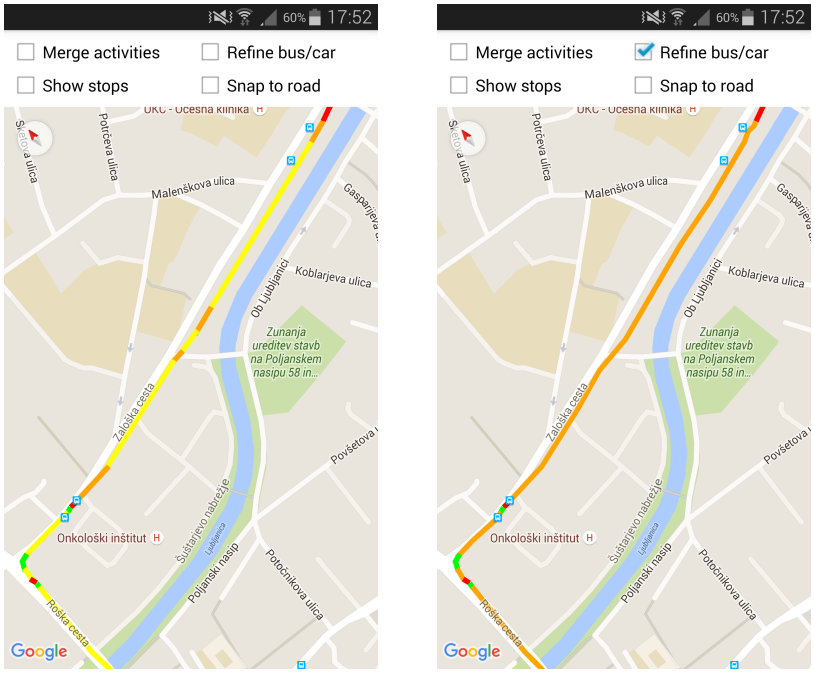


Slika 5.6: Primer poravnave poti s cestami.

5.4.3 Razločitev med avtobusom in avtomobilom

Naknadna razločitev med mestnim avtobusom in avtomobilom temelji na številu in lokaciji postankov na poti. Ker se razdalje med avtobusnimi postajališči po svetu v povprečju gibljejo med 100 in 400 metri, lahko sklepamo, da se bo avtobus na vsak prevožen kilometer ustavil vsaj na enem postajališču, tudi če kakšnega prevozi [17]. Pogoj za delovanje je vsaj en prevožen kilometer, skupna dolžina pa je izračunana kot seštevek razdalj med zaporednimi točkami na poti. Vsak postanek, ki je daljši od 10 sekund in se začne oz. konča z avtobusnim ali avtomobilskim segmentom, pride v poštev kot postajališče.

Za primerjavo z dejanskimi postajališči je uporabljen vmesnik Google Places API Web Service z zahtevkom Nearby Search, ki najde mesta okoli podane lokacije [5]. Gre za zahtevek HTTP GET v obliki `https://maps.googleapis.com/maps/api/place/nearbysearch/`



Slika 5.7: Primer naknadne razločitve med avtobusom in avtomobilom. Postanki (rdeči predeli) so blizu označenih postajališč, zato algoritem ugotovi, da gre za avtobus (oranžna barva).

`output?parameters`, kjer `output` pomeni želeno obliko odgovora (`json` ali `xml`), parametri pa so sledeči:

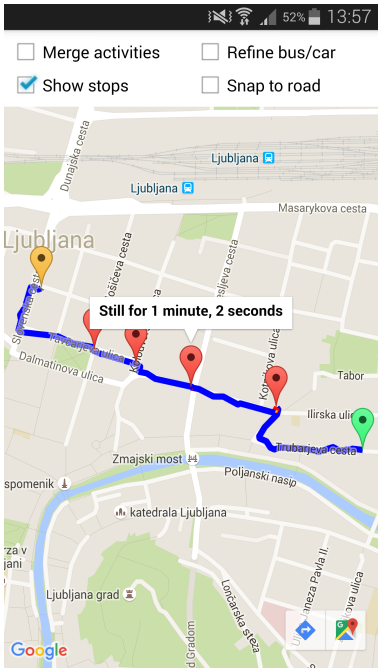
- `location` – zemljepisna širina in zemljepisna dolžina, ločeni z vejico,
- `radius` – radij območja okoli podane lokacije, v katerem naj se nahajajo rezultati, v tem primeru 30 metrov,
- `type` – tip iskanega mesta, v tem primeru `bus_station`,
- `key` – ključ za identifikacijo aplikacije.

Za komunikacijo HTTP je spet uporabljena knjižnica Volley, preko katere se pošljejo zahtevki z lokacijami vseh potencialnih postankov. Odgovor je JSON objekt s seznamom mest, ki ustrezajo omejitvam. Prazen seznam

pomeni, da v 30 metrih okoli podane lokacije ni nobenega avtobusnega postajališča. Če število ujemajočih se postajališč preseže določen delež postajališč na kilometer, se za prevozno sredstvo privzame mestni avtobus, v nasprotnem primeru pa gre za vožnjo z avtomobilom. Na sliki 5.7 je prikazan primer naknadne razločitve med vozili.

5.4.4 Prikaz postankov

Kot četrto možnost ima uporabnik na voljo še prikaz postankov, ki jasneje prikaže mirujoče segmente. Kot prikazuje slika 5.8, se na vseh lokacijah z mirujočo aktivnostjo postavijo markerji z informacijskimi okni, ki prikažejo dolžino postanka.



Slika 5.8: Primer prikaza postankov.

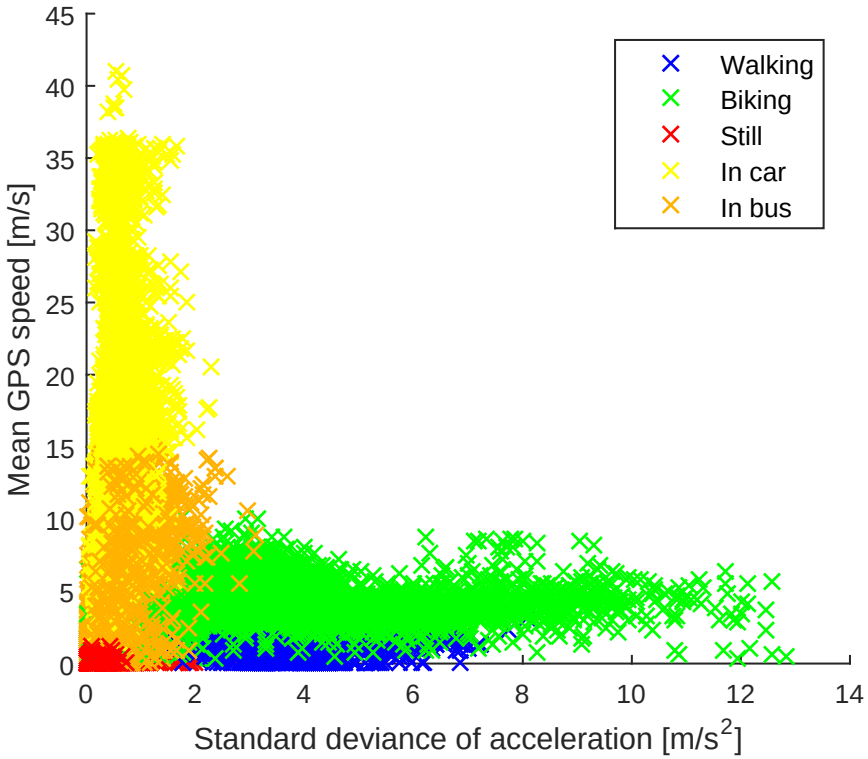
Poglavje 6

Rezultati

6.1 Analiza izbranih značilk

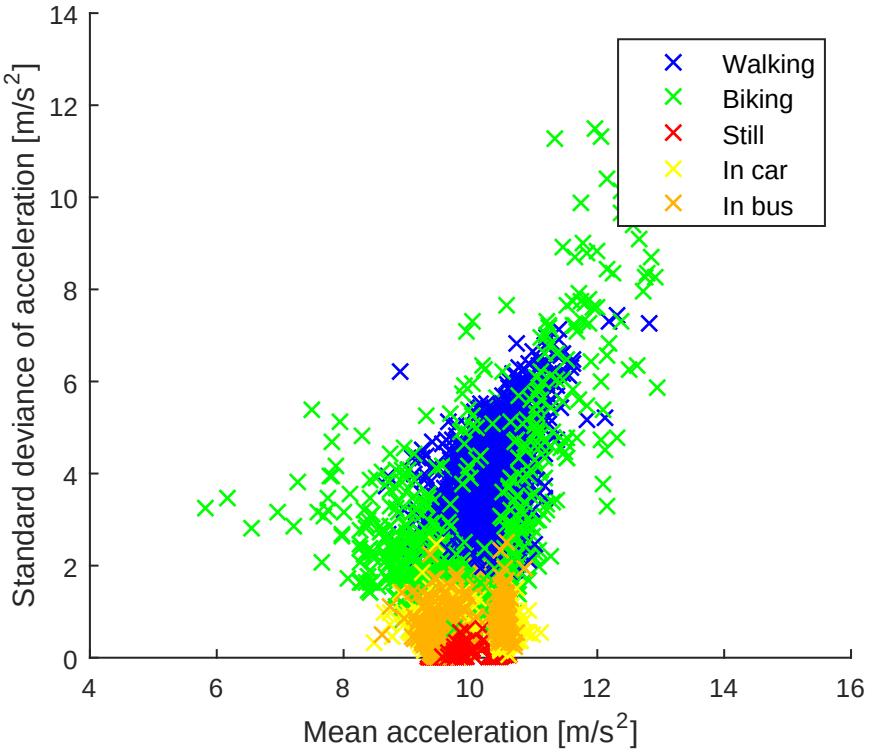
Podatki za učenje modelov so bili zbrani z dvema mobilnima napravama (Samsung Galaxy Note ter Samsung Galaxy Note 4) predvsem na območju Ljubljane. Narejenih je bilo 115 posnetkov, iz katerih je bilo s 4-sekundnim drsečim oknom izračunanih po 26911 primerov v obliki vektorjev značilk za modele s podatki GPS in 387 primerov za vsak model brez lokacijskih podatkov.

Izbira značilk omogoča dokaj zanesljivo ločevanje med razredi. Primerjavo med povprečno hitrostjo in standardnim odklonom ponazarja graf 6.1, kjer so izrisani vsi uporabljeni učni primeri s podatki GPS. Razvidno je širše območje, v katerem se razreda avtobusa in avtomobila prekrivata. Za klasifikacijo aktivnosti v tem območju tudi druge značilke ne zadoščajo, zato je problem rešen z naknadnim razločevanjem na osnovi postankov. Avtobus vseeno dosega nekoliko višje vrednosti za standardni odklon pospeška, nad hitrostjo 15 m/s pa se pojavljajo samo še primeri iz razreda avtomobila. Najvišje vrednosti standardnega odklona dosega kolesarjenje, sledi pa mu hoja, pri kateri je razvidna tudi nižja hitrost. Primeri mirujočega stanja se na obeh oseh gibljejo okoli ničle, odstopanja pa gre pripisati premikanju naprave in manjši natančnosti signala GPS.



Slika 6.1: Graf razmerja med povprečno hitrostjo in standardnim odklonom za primere iz razredov s podatki GPS.

Razločevanje med aktivnostmi brez podatkov GPS je nekoliko težje, kot prikazuje graf razmerja med standardnim odklonom pospeška in njegovim povprečjem 6.2. Pri kolesarjenju je razvidna večja razpršenost vrednosti povprečnega pospeška, dosežene pa so tudi najvišje vrednosti standardnega odklona. Hoja in mirujoče stanje imata podobno povprečno vrednost (v okolici gravitacijskega pospeška), razlikujeta pa se po standardnem odklonu, ki je pri hoji občutno višji. Z grafa je tudi jasno razvidno, da samo z uporabo povprečnega pospeška in njegovega standardnega odklona razreda avtobusa in avtomobila skoraj nista razločljiva.



Slika 6.2: Graf razmerja med standardnim odklonom pospeška in njegovim povprečjem za primere iz razredov brez podatkov GPS.

6.2 Določitev parametrov

Parametri za učenje modelov in klasifikacijo si bili določeni s pomočjo 3-kratnega prečnega preverjanja s tremi ponovitvami. Postopek je bil izvršen za vse kombinacije normaliziranih in nenormaliziranih podatkov z argumenti za regularizacijo z vrednostmi od 0 do 5 in stopnjo preslikave značilk z vrednostmi od 1 do 4. S primerjavo ugotovljenih napak so bili določeni optimalni argumenti za izračun parametrov.

Najboljši rezultati so bili pridobljeni z uporabo Newtonove metode, v redkih primerih pa je bila napaka gradientnega spusta nižja. Pri slednjem je pomembna uporaba ustreznega faktorja, ki ga je bilo potrebno določiti empirično, za optimalno delovanje pa morajo biti podatki normalizirani. Re-

zultati uporabe implementiranih algoritmov so bili v večini primerov boljši od vgrajene metode `fminsearch`, ki išče minimum funkcije napake, z ustreznimi argumenti pa so bili boljši tudi od metode `glmfit`. Ker slednja ne vsebuje regularizacije, je bila razlika najbolj očitna pri višjestopenjskih preslikavah značilk.

Tabeli 6.1 in 6.2 prikazujeta napako klasifikacije pri optimalnih argumentih za razrede z in brez podatkov GPS. Iz njiju izhajajo naslednje ugotovitve:

- Normalizacija podatkov je uporabna, če je učenje modelov opravljeno samo z gradientnim spustom, ki v nasprotnem primeru dosega slabše rezultate. Ker je bila uporabljena tudi Newtonova metoda, so bile napake na nenormaliziranih podatkih v vseh primerih za odtenek nižje.

- Najboljše rezultate dosega hoja v razredu brez podatkov GPS. Ker gre za značilen vzorec pospeškov brez večjih variacij, podatki GPS niso potrebni in lahko ob nižji natančnosti celo poslabšajo rezultat. Na drugem mestu je klasifikacija mirovanja, pri čemer je razred s podatki GPS nekoliko uspešnejši. Med postankom na poti je lahko mobilna naprava v uporabi in magnituda pospeška ni nujno nič, zaradi česar je podatek o hitrosti koristen. Sledi klasifikacija kolesarjenja, ki je brez podatkov GPS nekoliko uspešnejša. Tudi tu lahko zaradi podatka o hitrosti pride do zamenjave z avtobusom ali avtomobilom med speljevanjem. Slednja sta še posebej ob odsotnosti podatkov GPS težko ločljiva, kot potrjuje slika 6.2.

- Z izjemo normalizacije se optimalni parametri za učenje modelov razlikujejo glede na razred. Za klasifikacijo je bilo potrebno izbrati enoten set parametrov, zato je bil uporabljen najpogostejši par argumentov, kjer je stopnja preslikave značilk enaka 1 ter parameter za regularizacijo λ enak 0.

Razred	Napaka	Stopnja	λ	Normalizacija
Mirovanje	0,0020	1	0	ne
Hoja	0,0120	2	0	ne
Kolo	0,0270	2	2	ne
Avtomobil	0,0383	1	0	ne
Avtobus	0,0423	2	1	ne

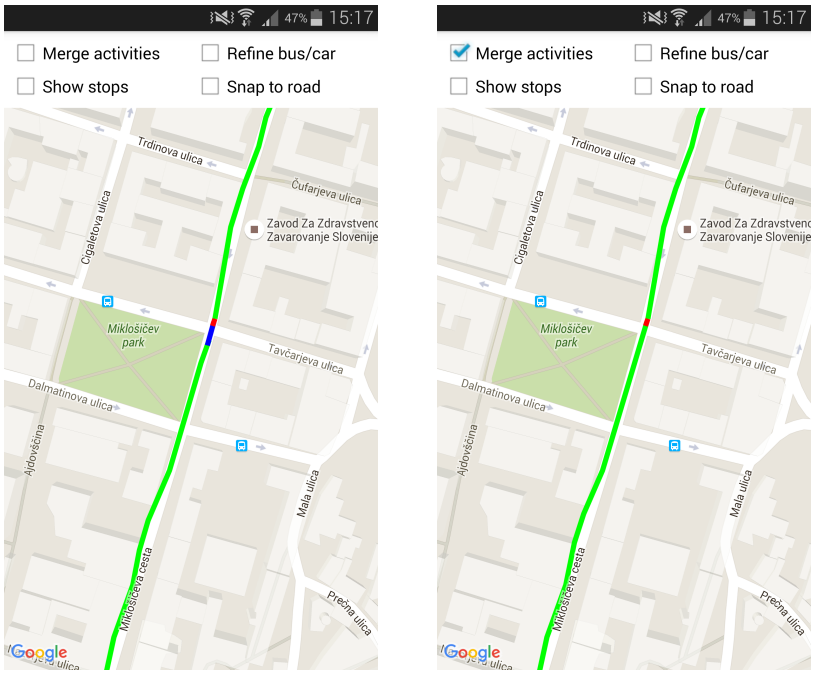
Tabela 6.1: Optimalni argumenti in pripadajoče povprečne absolutne napake klasifikacije za razrede s podatki GPS.

Razred	Napaka	Stopnja	λ	Normalizacija
Mirovanje	0,0045	2	4	ne
Hoja	0,0015	3	2	ne
Kolo	0,0191	1	0	ne
Avtomobil	0,0514	1	0	ne
Avtobus	0,0578	1	0	ne

Tabela 6.2: Optimalni argumenti in pripadajoče povprečne absolutne napake klasifikacije za razrede brez podatkov GPS.

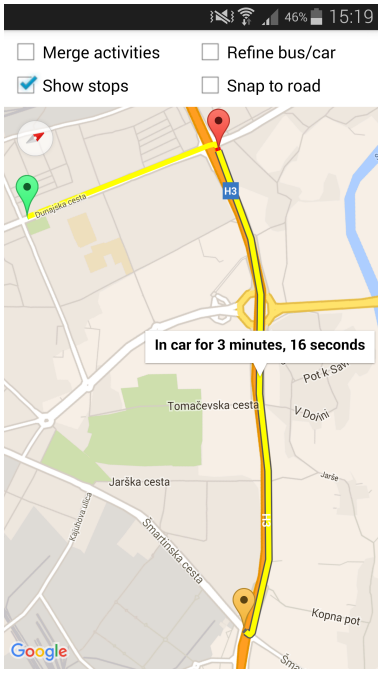
6.3 Empirične ugotovitve

Sprotna klasifikacija je v praksi večinoma zanesljiva, pojavljajo pa se nekatere težave. Hitra hoja se lahko ob nižji natančnosti GPS napačno klasificira kot kolesarjenje, enako pa se dogaja tudi ob speljevanju v avtomobilu ali avtobusu, kolesarjenje pa se med speljevanjem lahko napačno privzame za hojo. Naknadna korekcija z združevanjem osamelcev težavo reši, kot prikazuje slika 6.3.



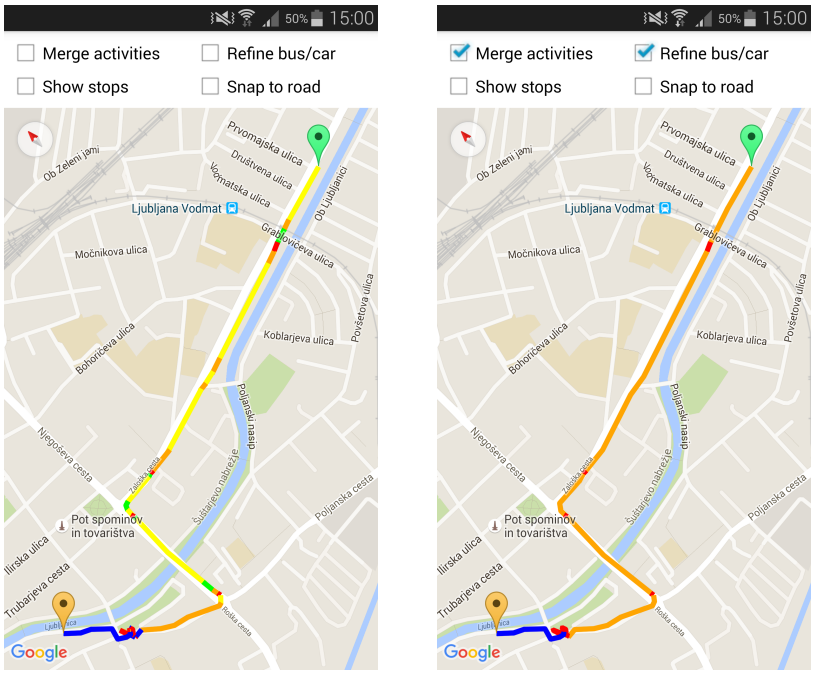
Slika 6.3: Primer vožnje s kolesom (zelena) z enim postankom (rdeča). Speljevanje je bilo napačno klasificirano kot hoja (modra), ki se na desni sliki po izbiri opcije združi v kolesarjenje.

Razločevanje med avtomobilom in avtobusom povzroča težave, ker so lahko njuni pospeški podobni. Zato velik faktor predstavlja hitrost. Pri visokih hitrostih (npr. na avtocestah in hitrih cestah) zagotovo ne gre za mestni avtobus in avtomobilska vožnja je klasificirana zanesljivo, kot prikazuje slika 6.4.



Slika 6.4: Primer vožnje z avtomobilom po hitri cesti. Klasifikacija je zaradi visokih hitrosti zanesljiva.

Pri hitrostih nad 50 km/h se za prevozno sredstvo pogosto privzame avtomobil, čeprav mestni avtobus dosega tudi višje hitrosti. Slika 6.5 prikazuje primer hoje in vožnje z avtobusom, kjer je bil začetni del vožnje klasificiran pravilno, potem pa je bil zaradi višje hitrosti (okoli 60 km/h) večinoma privzet avtomobil. Med speljevanjem in ustavljanjem so se pojavljali tudi osamelci, klasificirani kot kolesarjenje. Naknadna korekcija klasifikacije na podlagi postankov v kombinaciji z združevanjem osamelcev je težave v celoti odpravila.



Slika 6.5: Primer hoje (modra), postankov (rdeča) in vožnje z avtobusom (oranžna) z napačno klasificiranimi predeli kolesarjenja (zelena) in vožnje z avtomobilom (rumena). Desna slika prikazuje naknadno korekcijo klasifikacije.

Poglavje 7

Sklep

V sklopu diplomske naloge je bila razvita mobilna aplikacija, ki omogoča razpoznavanje prevoznih sredstev na osnovi modelov, naučenih z logistično regresijo. V ta namen je bila izdelana aplikacija za zbiranje podatkov, ki so bili uporabljeni za učno množico. Učenje je potekalo s pomočjo različnih parametrov in metod, po validaciji pa so bili za klasifikacijo uporabljeni najbolj uspešni modeli. Implementirana je bila tudi možnost naknadne korekcije napačno razpoznanih poti.

Izkazalo se je, da je klasifikacija načinov premikanja z binarno logistično regresijo dovolj uspešna in ni potrebe po bolj zapletenih algoritmih. Slabše se je odrezalo le razločevanje med avtomobilom in avtobusom, pri katerem bi za boljše sprotno delovanje potrebovali dodatne vhodne podatke. Problem je bil rešen naknadno s primerjavo lokacij postankov in avtobusnih postajališč.

Prostora za razširitve je kar precej. Lahko bi dodali modele za druga prevozna sredstva (npr. vlak) in za vhodne podatke uporabili dodatne senzorične vire (npr. iz športnih zapestnic ali pametnih ur), aplikacija pa bi lahko na osnovi zbranih posnetkov prikazovala prilagojene napovedi o predvidenem trajanju in najbolj primernem prevoznem sredstvu za neko pot. Snemanje bi lahko potekalo neprestano, poti pa bi se ločevale glede na definirane lokacije ali daljše mirovanje, da uporabniku ne bi bilo treba ročno zaganjati snemanja. Tako bi se samodejno ustvarjali koristni napotki, prilagojeni uporabniku.

Literatura

[1] R. Bajaj et. al., "GPS: location-tracking technology," *Computer*, zv. 35, št. 3, str. 92–94, 2002.

[2] L. Bao, S. S. Intille, "Activity recognition from user-annotated acceleration data," *International Conference on Pervasive Computing*, str. 1–17, 2004.

[3] Google. *Activity.* [Online]. Dostopno na: `https://developer.android.com/reference/android/app/Activity.html`

[4] Google. *Location Strategies.* [Online]. Dostopno na: `https://developer.android.com/guide/topics/location/strategies.html`

[5] Google. *Place Search.* [Online]. Dostopno na: `https://developers.google.com/places/web-service/search`

[6] Google. *Services.* [Online]. Dostopno na: `https://developer.android.com/guide/components/services.html`

[7] Google. *Snap to Roads.* [Online]. Dostopno na: `https://developers.google.com/maps/documentation/roads/snap`

[8] Google. *Storage Options.* [Online]. Dostopno na: `https://developer.android.com/guide/topics/data/data-storage.html`

[9] B. Hammack. *How a Smartphone Knows Up from Down.* [Online]. Dostopno na: `http://www.engineerguy.com/elements/videos/video-accelerometer.htm`

[10] T. Hastie et al., *The elements of statistical learning,* 2. izd. Stanford, CA: Stanford University, Dept. of Statistics, 2009.

[11] A. M. Khan, "Human Activity Recognition Using A Single Tri-axial Accelerometer", doktorska disertacija, Kyung Hee University Seoul, Korea, 2011.

[12] I. Kononenko, M. Robnik Šikonja, *Inteligentni sistemi.* Ljubljana: Založba FE in FRI, 2010.

[13] A. Ng. *CS229 Lecture notes.* [Online]. Dostopno na: `http://cs229.stanford.edu/notes/cs229-notes1.pdf`

[14] A. Ng. *Machine learning.* [Online]. Dostopno na: `http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=MachineLearning`

[15] B. Orel, *Osnove numerične matematike,* 3. izd. Ljubljana: Fakulteta za računalništvo in informatiko, 2004.

[16] C. R. Shalizi. (2016). *Advanced Data Analysis from an Elementary Point of View.* [Online]. Dostopno na: `http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/ADAfaEPoV.pdf`

[17] J. Walker. (2010). *Basics: The Spacing of Stops and Stations.* [Online]. Dostopno na: `http://humantransit.org/2010/11/san-francisco-a-rational-stop-spacing-plan.html`